HIERARCHICAL CONTROL FOR ROBOTS IN AN AUTOMATED FACTORY

J. S. Albus, C. R. McLean, A. J. Barbera, M. L. Fitzgerald

National Bureau of Standards
Washington, D.C. 20234

INTRODUCTION

The basic structure of a hierarchical control system is a tree, wherein each computational module has a single superior, and one or more subordinate modules. The top module is where the highest level decisions are made and the longest planning horizon exists. Goals and plans generated at this highest level are transmitted to the next lower level where they are decomposed into sequences of subgoals. In general, the decomposition at each level takes into account information derived from: (a) processed input data from sensors that measure the state of the environment, (b) reports from lower control levels as to the state of the control hierarchy itself, and (c) predictions (or expectations) generated by models, knowledge bases, or inference engines.

At each level, input commands from the next higher level are decomposed into sequences of output sub-commands to the next lower level in the context of the state of the environment, of the state of the control system, and the internal store of knowledge. At each level predictions and expectations are generated by the internal world model in the context of the state of the task, the goal of the system, and the best current hypothesis about the state of the environment. Also at each level, processed signals from the environment are compared against expectations from the world model. Correlations are computed and differences measured between observation and expectation. A high degree of correlation indicates that the task is proceeding according to plans, and expectations are being met. Differences represent error signals which can be used by the task decomposition module either to modify behavior, or change expectations so that the task goal is successfully accomplished. At the highest level, the input command represents the ultimate goal of the entire organism. At the lowest level, output drive signals are computed and sent to the physical actuators.

A hierarchical architecture for a factory control system is shown in Figure 1. A single chain of command from the bottom to the top of such a hierarchy is outlined by the dotted line in Figure 1. This chain of command can be further segmented, as shown in Figure 2, into three separate hierarchies: (1) a goal, or task decomposition, hierarchy (H); (2) a feedback processing hierarchy (G); and (3) a world model hierarchy (M). This has been discussed in a number of previous papers [2,3,4,5].

In general, each of the hierarchical levels shown in Figure 1 can be further partitioned into sublevels. For example the control hierarchy for a machining work station robot shown in Figure 2, decomposes the equipment level into three sublevels. An assembly robot might decompose the equipment level into four or more sublevels. The number of sublevels required depends on the complexity of the tasks that must be decomposed at that level.

At all levels, the H, G, and M modules are concurrent processes produced by real-time programs executing simultaneously in each module. Perhaps the simplest way to treat this conceptionally is to model each of the modules in the hierarchy as a finite-state automaton. The state-graph describing the activity of the entire hierarchy can then be described by a Petri diagram [1].

For each module in this architecture there are three concepts of time: the planning horizon, the response time, and the cycle time. The planning horizon is the interval over which a control module plans into the future. The response time is the delay between a change in a module's input and the generation of a new output. The cycle time is the period between sampling the input variables. In general, the response time will be slightly longer than the cycle time, and the planning horizon will be many times longer than the response time.

The response time of the finite-state automata at each level depends on the requirements for stability and dynamic response at the respective levels. The response time requirement is shorter at the lower levels, but the complexity of the control computations is less. The response time is longer at the higher levels, and the complexity of the computations is greater. Thus, the total computational power required at any level of the hierarchy is more or less constant.

Communication between the various modules in such a system can be accomplished by writing messages in a data base which is common to all modules which either compute or make use of those messages. Each message area (or mailbox) within the data base can be restricted so that only one system may write into it, although many can read its contents. If the cycles of the state-clock at all levels are synchronized, information transfer into and out of the common data base will occur at predictable time increments and each message can carry a time tag.

TASK DECOMPOSITION

In the robot control system architecture shown in Figure 2 the bottom (or first) level of the task decomposition hierarchy is where coordinate transforms and servo computations are made, and all joint motions are scaled to hardware limits on velocity and force.

At the second level, elemental movements such as <REACH TO (A)>, <GRASP>, <LIFT>, <ORIENT ON (B)>, <MOVE TO (X)>, <RELEASE>, etc. are decomposed into force and velocity trajectories in a convenient coordinate system. That coordinate system may be defined in the robot's work space, in the part, or in a coordinate frame in the robot's gripper.

At the third level, simple tasks such as <FETCH (A)>, <MATE (B) TO (A)>, <LOAD TOOL (C) WITH PART (D)>, etc. are decomposed into elemental movements which can be interpreted by the second level.

In the Automated Manufacturing Research Facility (AMRF) currently under construction at the National Bureau of Standards [8], a machining workstation robot will receive input commands to the third level of its hierarchical control system from a WORKSTATION CONTROLLER, which is the fourth level in the robot's hierarchy, as indicated in Figure 2.

A typical machining workstation in the AMRF will consist of a robot, a machine tool, a work tray buffer, and several tools and sensors that the robot can manipulate. Trays of parts and tools will be delivered to the workstation by a robot cart. The workstation controller will be given commands consisting of lists of operations to be performed on the parts in the trays. It is the task of the workstation controller to generate a sequence of simple task commands to the robot, the machine tool, and any other system under its control so that the set of operations specified by its input command list are carried out in an efficient sequence. For example, the workstation controller may generate a sequence of simple task commands to the robot to set up the clamping fixtures for the first part; to the machine tool to perform the specified machining operations; to the robot to modify the clamping fixtures for the next job, etc. The planning horizon for the workstation may vary from an hour or two up to about a day, depending on the complexity and number of parts that are being processed.

The fifth level of the robot control hierarchy in Figure 2 is the CELL CONTROLLER which is responsible for managing the production of a batch of parts within a particular group technology part family. The task of the cell is to group parts in trays and route the trays from one workstation to another. The cell generates dispatching commands to the material transport workstation to deliver the required tools, fixtures, and materials to the proper machining workstations at the appropriate times. The cell must have planning and scheduling capabilities to analyze the process plans for each part, the tooling and fixturing requirements, and the machineability time estimates for each operation. It will use these capabilities to optimize the make-up of trays and their routing from workstation to workstation. The planning horizon for the cell will depend on the size and complexity of the batch of parts in process, but may be on the order of a week.

The sixth level in the robot control hierarchy is the SHOP CONTROLLER which performs long term production planning and scheduling. It also manages inventory, and reports shortages in materials and tools to the next higher level (Facility Control) where orders are issued to outside vendors. The planning and scheduling functions are used to determine the workstation, robot, and material resources requirements for each cell. The shop then dynamically allocates workstations to, or reclaims them from the cells as necessary to meet the production schedule [7]. The control structure always satisfies the rules of a hierarchical tree at any instant in time, but the subtrees of the shop may shift from one moment to the next. For example, an individual robot may belong to Workstation #1 of Cell #26 at one moment, and to Workstation #16 of Cell #2 the next. The same logic can be applied to the passing of workstations between cells. This degree of flexibility becomes important in factories or construction sites where robots are mobile and rapidly move from one physical work site to another.

The seventh level is FACILITY CONTROL. It is at this level that engineering design is performed and the process plans for manufacturing each part, and assembling each system, are generated. Here also, management information is analyzed, materials requirements planning is done, and orders are processed for maintaining inventory. Because of the very long planning horizons at this level in the control hierarchy, the activities of the facility control module are not usually considered to be a part of a real-time control system. However, in the context of hierarchical control with exponentially increasing time horizons at each higher level, these facility control activities can be integrated into the real-time control hierarchy of the manufacturing system.

FEEDBACK PROCESSING

Each level of the task decomposition hierarchy is serviced by a feedback processing module which extracts the information needed for control decisions at that level from the sensory data stream and from the lower level control modules. The feedback processing modules at each level detect features, recognize patterns, correlate observations against expectations, and format the results to be used in the decisions and computational procedures of the task decomposition modules at that level.

At the lowest level of the robot hierarchy, the feedback processing modules extract and scale joint positions and force and torque data to be used by the servo and coordinate transformation computations.

At the second level, touch and proximity data, and simple visual measurements of distance and positions of grip points are extracted from the sensory input to be used in computing trajectory end points.

At the third level the three dimensional positions of visual features such as edges, corners, and holes are computed and combined to determine the position and orientation of surfaces and volumes of objects. Identities of objects may also need to be computed (or recognized) in order to generate the reaching and grasping commands at this level.

At the fourth (WORKSTATION) level, relationships between various objects need to be determined, in order to sequence simple task commands.

At the fifth (CELL) level, the location and composition of trays of parts and tools and the length of queues of parts needs to be determined. This may be derived from sensors which read coded tags on trays, or may be inferred from sensory input from lower level sensors on the robot or in the workstation.

At the sixth (SHOP) level, the condition of machines, tools, and the amount of inventory on hand must be determined in order to generate schedules, allocate resources, and evaluate and set priorities for production.

At the seventh (FACILITY) level, the requirements for changes in part design, or in process plans need to be recognized in order to make engineering changes, or redesign parts or processes.

THE WORLD MODEL

The world model hierarchy, made up of M modules in Figure 2, consists of a knowledge base containing all the information currently known about the task, the parts, or the workplace. The M modules also contain procedures that allow them, based on the state of the task and other contextual information from various places in the hierarchy, to compute expectations and predictions about what the sensory data to the corresponding G module should be. This allows the G modules at each level to compare expectations with observations, and to measure both the degree of correlation and the degree of difference. A strong degree of correlation means that the proper model is being matched with the incoming sensory data. It means that the observed object or situation has been correctly recognized, and that information contained in the model can be safely used for decision making even though it may not be directly observable by the sensory system.

A large degree of difference between expectations generated by the model and observations derived from sensors means that either an incorrect choice of models has been made, or the model has not been correctly transformed spatially or temporally so as to generate the proper set of expected feature relationships, or that the incoming sensory data is too noisy, or is being improperly processed and filtered. In this case, the computational problem for the task decomposition module is to decide which type of error is being encountered and what is required to remedy the discrepancy. In general, this type of problem can be solved either by a set of situation/action rules of an expert system, or a set of heuristic search procedures.

At low levels, the world model contains dimensional information describing the shapes and sizes of parts. The M modules use this to generate expected positions of image features such as edges, corners, holes, and surfaces.

At the facility control level the model contains information about machining processes, material properties, shop processing capabilities, and expected lead times for procurements which can be used to compute estimated completion times for various production plans.

At the shop level, the world model contains information about machine capabilities, machineability of materials, tool life, and inventory levels and is able to simulate the performance of various cell configurations.

At the cell level, the model contains information about workstation task times, and is able to simulate the performance of various hypothetical task sequences.

At the workstation level, the world model contains knowledge of tray layouts including the names of parts and their approximate positions, orientations, and relationships such as on-top-of, underneath, stacked N-deep, leaning-against, etc.

At the simple task level, the model contains knowledge of the geometrical size and shapes of three dimensional objects such as parts and tools and the relationships between coordinate systems based in the work space and the robot. These can be used to generate expected positions and orientations of three dimensional objects in a robot or machine tool coordinate system.

At the elemental move level, the model is able to generate expected positions and orientations of specific features of parts and tools, such as edges, corners, surfaces, holes, and slots [6].

At the coordinate transformation and servo level, the model generates windows or filter functions that are used to screen and track the incoming raw data stream.

PLANNING IN A HIERARCHICAL CONTROL SYSTEM

There can be three different modes of operation of a hierarchical control structure such as shown in Figure 2.

The first is the control execution mode. In this mode the task decomposition hierarchy is used to decompose tasks into subtasks in the generation of behavioral action. The world model hierarchy is used to generate expectations based on the state of the task. The sensory processing hierarchy compares these expectations against observed sensory data in evaluating the results of actions. Processed feedback information is used to servo the control system to successfully accomplish its goal.

A second mode of operation is the sensory recognition mode. In this mode the control hierarchy is used to generate hypotheses which drive the world model hierarchy in the creation of expectations. The sensory processing hierarchy compares these expectations against observed data in order to recognize and evaluate objects, situations, and events.

A third mode of operation is the planning mode. In this mode the control hierarchy is used to generate hypothesized tasks and task decompositions. These drive the world model hierarchy to create expected hypothetical results. The sensory processing hierarchy evaluates these hypothetical results and assigns a desirability (or cost) value to the hypothesized task decompositions. By this mechanism a search can be conducted over the space of potential task decompositions to find an optimal coarse of action prior to the initiation of behavioral output.

At each level of the control hierarchy, a computing module such as shown in Figure 3 can be used to execute the production rules that encode the control program at that level. A state-graph (or Petri diagram) corresponding to the rules in the state-transition table is the analog of the flow chart of a procedural program for the task decomposition module [1]. The left-hand side of the table consists of all the command, internal state, and feedback inputs that can be encountered at any tick of the state-clock. The right-hand side contains an output command (and/or a pointer to a procedure which computes an argument which becomes part of the output command) to the next lower level. It also contains a next internal state, and a report to the next higher level, or to other modules at the same level.

At the lowest hierarchical level, the left-hand side of the state-transition table consists of variables which select the type of coordinate transformation required and the type of servo computations needed.

At the second level, the left-hand side consists of variables which define the type of trajectories to be generated. The right-hand side contains pointers to procedures that compute forces, positions, accelerations, and velocities in the appropriate coordinate systems.

At the third level, the left-hand side consists of variables which specify the state of the environment as reported by sensors, and the right-hand side the names of appropriate elemental movements to be made for each state. Pointers to procedures are used to compute arguments and modifiers.

At the higher levels, the state-tables may be compared to production rules in expert systems. Procedures that are invoked by these state-tables may consist of heuristic search algorithms or linear programming techniques for generating plans, schedules, etc.

The response time and cycle time requirements grow longer for the finite-state automata at the upper levels of the hierarchy. Thus, the amount of computing power needed in the execution mode to execute state-transition tables decreases at higher levels in the hierarchy. On the other hand, there is much more need for planning at the upper levels. For example, the types of control decisions required at the upper levels of the factory control system shown in Figure 2 typicaly involve planning algorithms. The hierarchical control system proposed here thus provides the required planning capability. The upper levels of the control hierarchy can use the excess execution mode computing power to operate in the planning mode.

CONCLUSIONS

There appear to be a number of advantages of the table-driven hierarchical control system described above. The first is that it partitions the problem into simple, well-defined modules with clearly specified inputs, outputs, internal states, and rules for state-transitions. The control problem is partitioned vertically with respect to task complexity and abstraction, horizontally with respect to function (such as task decomposition, sensory processing, and world modeling), and along the time axis by the use of a state-clock. This simplifies the design and eases the synchronization of simultaneous processes in the many different computing modules.

The second advantage is that it facilitates the handling of error conditions. If additional states need to be defined to deal with unanticipated error conditions, they can simply be inserted into the state-graph and added to the state-transition table. There is very little interaction with other parts of the program code. A simple interpreter can insert the additional steps into the state-tables as fast as they are entered from the keyboard. Program executions can then continue from the point where the error was encountered.

Third, this approach formalizes the control problem into a very orderly structure. Each line in the state-transition table for any module is an IF/THEN production rule. <IF (the command is such, and the state is so, and the feedback conditions are thus) / THEN (the output is whatever is stored on the right hand side of the table, and the system steps to the indicated next state)>. The addition of each node or edge to the state-graph, and the corresponding lines added to the state-transition table is the equivalent of the addition of a new chunk of knowledge about how to deal with a specific control situation at a particular point in a problem domain at a unique phase in the task execution. This system architecture thus bridges the gap between servomechanisms and finite-state automata at the lower levels and expert system technologies at the upper levels. The fact that even very powerful expert systems typically do not contain more than a thousand production rules suggests that the size of the state transition tables for sensory interactive robot control systems will not become excessive, even at the higher levels.

Fourth, it allows the easy insertion of conditional tests of new sensory or feedback data. For example, if a new touch sensor is added, it is possible to merely insert a new column in the feedback portion of the state-transition table, and a new line, or lines, in the state-transition table for each of the new edges to be added to the state-graph.

Fifth, it facilitates debugging. Each state of the system is formally identified and the set of conditions that lead to and from that state are clearly specified. Diagnostic routines can be used to read these states from common memory and stop the state-clock when necessary. This makes it easy to perform traces, to set break points, and to reason backwards from error states. The system is completely deterministic and errors in logic are simple to reconstruct. Program bugs are therefore relatively easy to locate and correct.

Sixth, it makes it possible to build teaching and learning capabilities into sensory-interactive robot control systems. Simple programs with few or no error conditions can be defined first by state-graph flowcharts. The system can then be run until it encounters unanticipated problems or undefined conditions which cause an <ERROR STOP>. Then each problem condition can be dealt with specifically by adding lines to the state-transition table to address that particular problem state. Eventually the entire space of problem states will contain programmed solutions, and the frequency of <ERROR STOP>'s will decline. The rule-based approach facilitates the integration of an expert system which would have the capability to use its higher "learning" rules to modify the world model, sensory processing, or decision structures, as experience on different problem states is acquired.

There are, of course, many unanswered questions regarding this approach. The hierarchical control system architecture described in this paper is still largely a theoretical construct. There are many problems that remain to be addressed and many of the details of this method will undoubtedly change as more experience is acquired during the construction and testing of the experimental system.

BIBLIOGRAPHY

1. Albus, J.S., Barbera, A.J., Fitzgerald, M.L. (1982). Programming a Hierarchical Robot Control System. Proc. 12th International Symposium on Industrial Robots, Paris, France, 9-11 June 1982.

2. Albus, J.S., Barbera, A.J., Fitzgerald, M.L. (1981a). Hierarchical Control for Sensory Interactive Robots. Proc. 11th International Symposium on Industrial Robots, Tokyo, Japan, 7-9 October 1981.

3. Albus, J.S., Barbera, A.J., Fitzgerald, M.L., Nashman, M. (1981b). Sensory Interactive Robots. Proc. of 31st General Assembly, International Institution for Production Engineering Research (CIRP), Toronto, Canada, September 1981.

4. Albus, J.S., Barbera, A.J., Nagel, R.N., (1981c). Theory and Practice of Hierarchical Control. Proc. 23rd IEEE Computer Society International Conference, September 1981, 18-39.

5. Barbera, A.J., Fitzgerald, M.L., Albus, J.S. (1982). Concepts for a Real-Time Sensory-Interactive Control System Architecture. Proc. 14th Southeastern Symposium on System Theory, Blacksburg, Virginia, April 1982, 121-126.

6. Kent, E.W. "A Hierarchical, Model-Driven, Vision System for Sensory-Interactive Robotics." Compsac 82, Chicago, IL, November 11, 1982.

7. McLean, C.R. (1982). The Virtual Manufacturing Cell. Proc. 4th IFAC/IFIP Symposium on Information Control Problems in Manufacturing Technology, Gaithersburg, Maryland, 26-28 October 1982.

8. Simpson, J.A., Hocken, R.J., Albus, J.S. (1982). The Automated Manufacturing Research Facility of the National Bureau of Standards. Journal of Manufacturing Systems, Society of Manufacturing Engineers 1:(1)17-31, 1982.

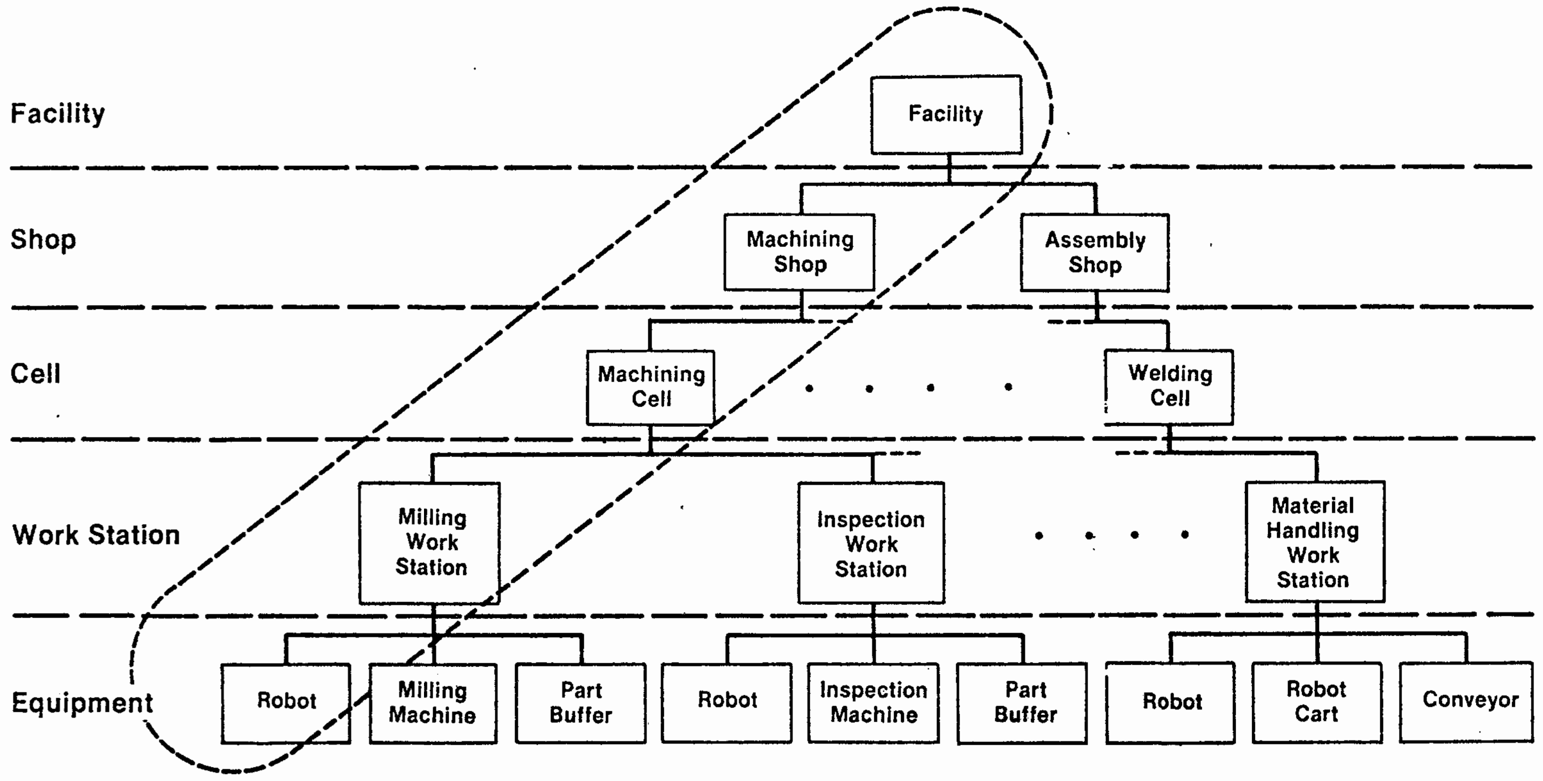

Figure 1. A hierarchical control system architecture for integration of robots, machine tools, and material transport facilities into an automatic factory. The chain of command enclosed in the dotted lines is detailed in Figure 2.

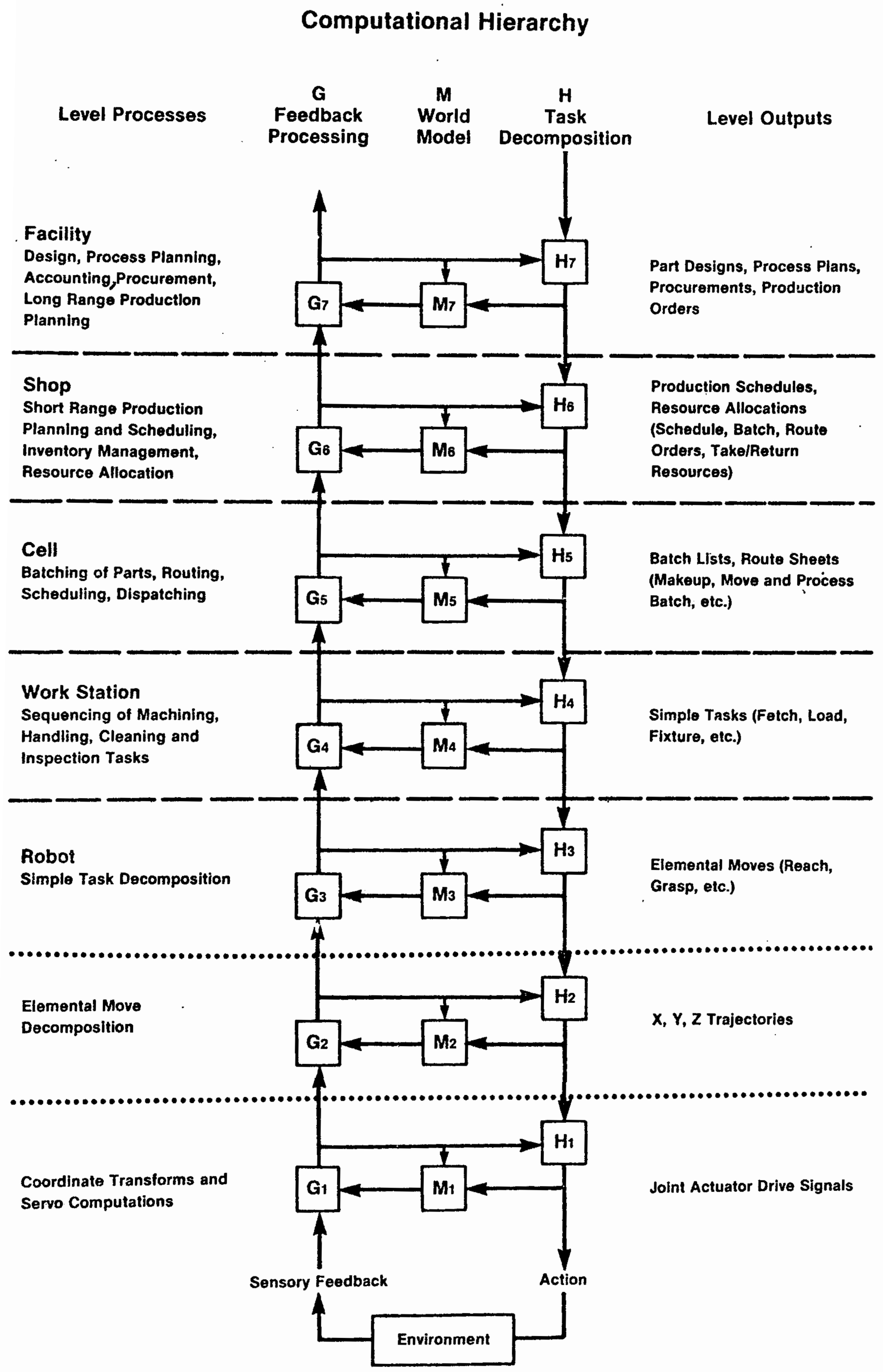
Computational Hierarchy
Level Processes
G Feedback Processing
M World Model
H Task Decomposition
Level Outputs
Facility
Design, Process Planning, Accounting, Procurement, Long Range Production Planning
Part Designs, Process Plans, Procurements, Production Orders
Shop
Short Range Production Planning and Scheduling, Inventory Management, Resource Allocation
Production Schedules, Resource Allocations (Schedule, Batch, Route Orders, Take/Return Resources)
Cell
Batching of Parts, Routing, Scheduling, Dispatching
Batch Lists, Route Sheets (Makeup, Move and Process Batch, etc.)
Work Station
Sequencing of Machining, Handling, Cleaning and Inspection Tasks
Simple Tasks (Fetch, Load, Fixture, etc.)
Robot
Simple Task Decomposition
Elemental Moves (Reach, Grasp, etc.)
Elemental Move Decomposition
X, Y, Z Trajectories
Coordinate Transforms and Servo Computations
Joint Actuator Drive Signals
G7 M7 H7
G6 M6 H6
G5 M5 H5
G4 M4 H4
G3 M3 H3
G2 M2 H2
G1 M1 H1
Sensory Feedback
Action
Environment

Figure 2. The computational hierarchy for a robot in a machining workstation. This hierarchy corresponds to the chain of command enclosed in dotted lines in Figure 1.

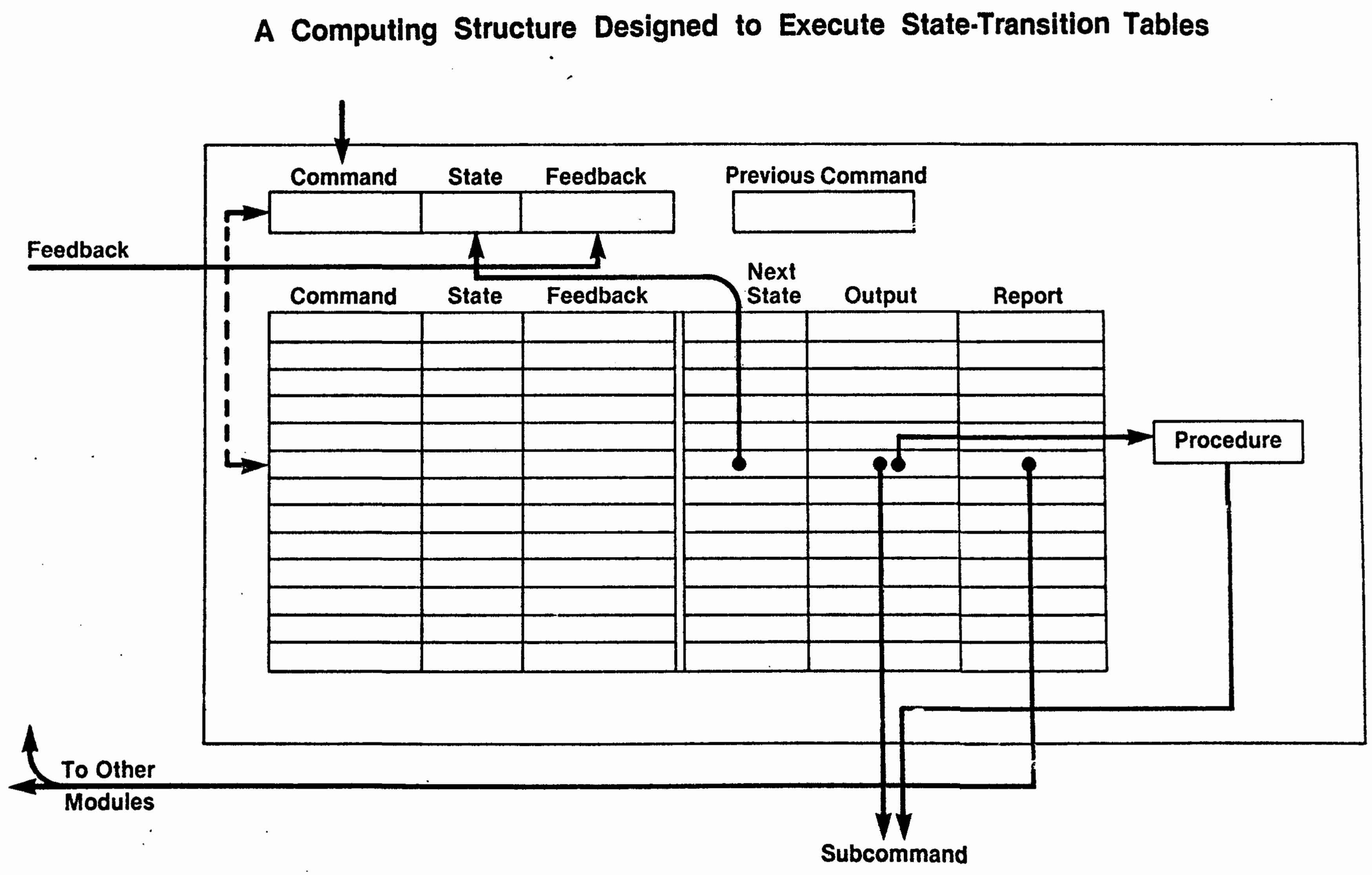

Figure 3. A computing structure to execute state-transition tables.